Multi-Analyst Differential Privacy for Online Query Answering

David Pujol
Duke University
dpujol@cs.duke.edu

Albert Sun
Duke University
albert.sun310@duke.edu

Brandon Fain
Duke University
btfain@cs.duke.edu

Ashwin Machanavajjhala
Duke University
ashwin@cs.duke.edu

ABSTRACT

Most differentially private mechanisms are designed for the use of a single analyst. In reality, however, there are often multiple stakeholders with different and possibly conflicting priorities that must share the same privacy loss budget. This motivates the problem of equitable budget-sharing for multi-analyst differential privacy. Our previous work defined desiderata that any mechanism in this space should satisfy and introduced methods for budget-sharing in the offline case where queries are known in advance.

We extend our previous work on multi-analyst differentially private query answering to the case of online query answering, where queries come in one at a time and must be answered without knowledge of the following queries. We demonstrate that the unknown ordering of queries in the online case results in a fundamental limit in the number of queries that can be answered while satisfying the desiderata. In response, we develop two mechanisms, one which satisfies the desiderata in all cases but is subject to the fundamental limitations, and another that randomizes the input order ensuring that existing online query answering mechanisms can satisfy the desiderata.

PVLDB Reference Format:
David Pujol, Albert Sun, Brandon Fain, and Ashwin Machanavajjhala. Multi-Analyst Differential Privacy for Online Query Answering. PVLDB, 16(4): 816 - 828, 2022.
doi:10.14778/3574245.3574265

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/yikai-wu/Multi-Analyst-DP/tree/asun-online-version.

1 INTRODUCTION

Analysis of sensitive information about individuals is essential for many research tasks. However since such analyses are often made public this may come at the cost of individual privacy [10, 16, 23]. Differential Privacy (DP) [4, 6] is often considered the gold standard of privacy protection. Differentially Private mechanisms often add randomized noise to data to hide the presence of an individual record while still preserving aggregate information. However, due to the fundamental law of information recovery [3], answering an unbounded amount of queries, even under Differential Privacy will eventually allow an attacker to accurately reconstruct the underlying dataset. Because of this, data curators must bound the amount of information released. In differential privacy this is captured by setting a bound on the privacy loss parameter, ϵ, often called the privacy loss budget or just simply the privacy budget. This turns any query answering problem into a resource allocation problem. Given a fixed ϵ how much of the privacy budget should be spent on each query and how will this budget be allocated across the interests of multiple stakeholders?

We study the Multi-Analyst Differential Privacy problem introduced in [20]. In this setting, there are multiple stakeholders all interested in a single dataset. The data curator must answer all of the analysts' queries while limiting the total privacy budget across all analysts to be bounded by ϵ. Each analyst is entitled to a specific share of the privacy budget. Since the privacy budget is finite and must be spent to answer queries, we treat it as a fundamental system resource to be distributed across the analysts.

In order to ensure acceptable utility for all analysts, [20] considers several additional desiderata that any mechanism in this space should satisfy. They introduce the hypothetical collective where an analyst has the choice of either answering their queries independently with their own share of the privacy budget or joining a group of other analysts and sharing their privacy budgets under a joint mechanism. First, the mechanism must satisfy the **Sharing Incentive**, meaning that the joint mechanism must offer each analyst at least as much utility as they would receive if they took their share of the privacy budget and answered their queries independently. This protects analysts joining the mechanism ensuring that their utility can only increase by so doing. A mechanism should also satisfy **Non-Interference** or the stronger property of **Analyst Monotonicity**, both stating (at a high level) that the addition of a new analyst to the collective (along with their respective resources) must not cause any analyst already in the collective to suffer a loss in utility. This protects analysts already participating in the joint mechanism, ensuring that the addition of new participants can only increase their utility.

Previous work [20] considers the offline case where all queries are presented ahead of time and a mechanism outputs all query answers at once. This offline query answering model is equivalent to one time data release such as the data products released by the US Census Bureau [17]. In this paper, we consider the extension to the online query answering model, where queries are asked one at a time without prior knowledge of the entire query sequence.

Proceedings of the VLDB Endowment, Vol. 16, No. 4 ISSN 2150-8097.
doi:10.14778/3574245.3574265

1.1 Contributions

We consider the extension to the Online setting where queries are asked one at a time and the number of queries being asked is not known by the mechanism. This allows for cases where the workload is not known in advance and queries may be sporadic. Our contributions are as follows.

- In Section 3 we extend the multi-analyst differentially private query answering problem to the online setting where queries are in the form of an ordered sequence and must be answered one at a time.
- We show experimentally in Section 4 that existing online mechanisms fail to satisfy the sharing incentive or non-interference and the disparities between analysts grows as the distribution of queries becomes skewed towards one analyst.
- We prove Theorem 6 which states that there is a fundamental limit to the number of queries a mechanism can answer while still satisfying the sharing incentive and that this limit scales polynomially in the number of analysts.
- On the positive side we provide two solutions to the online multi-analyst differential privacy problem. First we introduce a new mechanism in Section 6 called **Seeded Cache and Reconstruct**, an online multi-analyst query answering algorithm that satisfies differential privacy, sharing incentive, analyst monotonicity, and non-interference. This mechanism is subject to the upper bound on queries implied by Theorem 6 but requires no modification to the online query answering model.
- In Section 7 we introduce the **Query Scheduler**, a method of circumventing the limitations of Theorem 6 while still satisfying the sharing incentive by enforcing a structure on the order in which analysts ask queries. We show, by a reduction to the well known Counting Coupons problem, that by randomly selecting an analyst to answer one of their queries at each time step any efficient-enough algorithm satisfies the sharing incentive.
- Finally we show that this result generalizes to cases where the analysts are not chosen exactly uniformly at random and the privacy budget shares are not exactly equal.

2 BACKGROUND

Data Representation Like previous work [11, 14, 15, 18], we consider databases where each individual corresponds to exactly one tuple. The algorithms considered use a vector representation of the database denoted $\boldsymbol{x}$. More specifically, given a set of predicates $\mathcal{B} = \{\phi_1 \dots \phi_k\}$, the original database D is transformed into a vector of fractional counts $\boldsymbol{x}^D$ where $\boldsymbol{x}^D_i$ is the fraction of records in D which satisfy ϕ_i. We denote the length of the data vector as $|x|$ (often referred to as the dimension of the database), the number of individuals in the database n, and we will use the notation $\boldsymbol{x}$ to refer to the vector form of database D.

Predicate counting queries are a versatile and powerful class of queries that count the number of tuples satisfying a logical predicate. A predicate corresponds to a condition in the **WHERE** clause of an SQL query. A predicate counting query is one of the form **SELECT Count (*) FROM R WHERE** ϕ. Workloads of counting queries can express a rich and powerful set of queries such as histograms, range queries, marginals, and datacubes. Like databases, a predicate counting query can be represented as a $|x|$-length vector $\boldsymbol{q}$ such that the answer to the query is $\boldsymbol{q}^T \boldsymbol{x}$.

Differential Privacy [4, 6] is a formal model of privacy that guarantees each individual that any query computed from sensitive data would have been almost as likely as if the individual had opted out. More formally, Differential Privacy is a property of a randomized algorithm which bounds the ratio of output probabilities induced by changes in a single record.

Definition 1 (Differential Privacy). *A randomized mechanism $\mathcal{M}$ is (ϵ,δ)-differentially private if for two neighboring databases D, and D' which differ in at most one row, and any outputs $O \subseteq Range(\mathcal{M})$: the following holds:*

$$\Pr[\mathcal{M}(D) \in O] \leq \exp(\epsilon) \times \Pr[\mathcal{M}(D') \in O] + \delta$$

The parameter ϵ often called the privacy budget quantifies the privacy loss. δ can be seen as the probability of catastrophic failure, for this reason, δ is usually set to be negligibly low.

The Laplace Mechanism is a differentially private primitive which is utilized in many differentially private mechanisms.

Definition 2 (Laplace Mechanism). *Given a query vector q, the randomized algorithm which outputs the following vector is ϵ-differentially private [6].*

$$q\boldsymbol{x} + \text{Lap}\left(\frac{\|q\|_1}{\epsilon}\right)$$

Where $\|q\|_1$ is the maximum L1 column norm of q, often called the sensitivity of a query, and Lap(σ) denotes a sample from a Laplace distribution with mean 0 and scale σ. While the sensitivity of a query determines how much noise is necessary, for simplicity of analysis we will consider in this work only queries of sensitivity 1. Any linear counting query can be reduced to a sensitivity of 1 by multiplying it with a normalizing constant. Differentially private releases compose with each other in that if there are two private releases of the same data with two different privacy budgets the amount of privacy lost is equivalent to the sum of their privacy budgets. More formally we have the following.

Theorem 1 (DP composition [6]). *Let $\mathcal{M}_1$ be an ϵ_1-differentially private algorithm and $\mathcal{M}_2$ be an ϵ_2-differentially private algorithm. Then their combination defined to be $\mathcal{M}_{1,2}(x) = (\mathcal{M}_1(x), \mathcal{M}_2(x))$ is $\epsilon_1 + \epsilon_2$-differentially private*

This allows the privacy budget to be split across multiple mechanisms and have the overall privacy loss quantified across all the mechanisms. In this way, the privacy budget acts like a finite resource that can be spent on various individual tasks. **Private Multiplicative Weights** [11] (PMW) is an online differentially private mechanism that creates a synthetic database as queries are being answered. At each time step if the synthetic database can be used to answer an individual query with high enough accuracy then the synthetic database is queried, resulting in no privacy loss. If the synthetic database cannot be queried accurately then the true database is queried using the Laplace Mechanism and the noisy answer is then used to update the synthetic database. PMW is (ϵ, δ)-differentially private and satisfies (α, β, k) -

accuracy, meaning that it can answer k queries with error under $\alpha = O\left(\frac{((\log(d)+\log(1/\beta))\log^{1/4}(|D|)\log(1/\delta)}{\epsilon\sqrt{n}}\right)$ with probability $1-\beta$.

The Matrix Mechanism [15, 18] is an offline query answering mechanism for answering workloads of queries denoted $\boldsymbol{W}$. The mechanism creates an alternative "strategy workload" denoted $\boldsymbol{A}$ to answer directly and reconstructs the queries in $\boldsymbol{W}$ from $\boldsymbol{A}$. The error of the matrix mechanism is as follows.

$$\text{Error}(\boldsymbol{W}, \boldsymbol{A}, \epsilon) = \frac{2}{\epsilon^2} \|\boldsymbol{A}\|_1^2 \|\boldsymbol{W}\boldsymbol{A}^+\|_F^2 \tag{1}$$

This will be necessary to evaluate the error of mechanisms later. We will also rely on results initially from [20] which are as follows.

Lemma 1. *For any workload matrix $\boldsymbol{W}$ and any strategy $\boldsymbol{A}$*

$$\left\|\boldsymbol{W}(\boldsymbol{D}\boldsymbol{A})^+\right\|_F \leq \left\|\boldsymbol{W}\boldsymbol{A}^+\right\|_F$$

where $\boldsymbol{D}$ is a diagonal matrix with all diagonal entries greater than or equal to 1 and $\boldsymbol{A}$ is a full rank matrix.

Lemma 2. *Let $\tilde{\boldsymbol{A}}$ be the original strategy matrix $\boldsymbol{A}$ with additional queries (rows) added to it. We can write this as a block matrix as $\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{C} \end{bmatrix}$ Where $\boldsymbol{C}$ are the additional queries. For any workload $\boldsymbol{W}$ and any strategy $\boldsymbol{A}$*

$$\left\|\boldsymbol{W}\tilde{\boldsymbol{A}}^+\right\|_F \leq \left\|\boldsymbol{W}\boldsymbol{A}^+\right\|_F$$

Pujol et al. [20] showed that when using the matrix mechanism reconstruction step, for any strategy matrix $\boldsymbol{A}$ the addition of more queries with additional privacy budget (Lemma 2) or additional privacy budget for existing queries (Lemma 1) will always result in a reconstruction with lower expected error. These are results that we rely on heavily in the construction of the Seeded Cache and Reconstruct mechanism in Section 6.

Counting Coupons For results in Section 7 we rely on a reduction to the classic combinatorial problem of Counting Coupons.

The traditional Counting Coupons problem has an urn with k unique coupon types from which we sample with replacement. The goal is to calculate T_k, the number of coupons that need to be sampled in expectation in order to have at least one of each coupon type. We can generalize this problem in two ways. First, suppose we have a vector $\vec{m}$ indicating we need at least m_i copies of type i. Second, suppose we have a vector $\vec{x}$ indicating that there are x_i coupons of type i in the urn. The more general problem is to find $T_k(\vec{m}, \vec{x})$, the expected number of samples with replacement needed to obtain m_i copies of each type i, given x_i of that type in the urn. We use a scalar for $\vec{x}$ or $\vec{m}$ to denote a length k constant vector of that value repeated.

We will rely on several results from the literature namely the following theorems.

Theorem 2 (Coupon Collecting with Non-Uniform Probability [22]). *Let k be the number of unique coupons, m be the vector representing the quotas for each coupon and p be the vector representing the number of each coupon in the urn. Let $P_k = \sum_{i=0}^{k} \vec{p}_i$ and let $S_{m_i}(t) = \sum_{k=0}^{m_i-1} \frac{t^k}{k!}$, the first m_i terms of the Taylor series expansion of e^t. Then the expected number of samples required to collect m_i coupons of each type denoted $T_n(\vec{m}, \vec{p})$ is as follows*

$$T_k(\vec{m}, \vec{p}) = X_n \int_0^\infty \left(1 - \prod_{i=1}^{n} (1 - S_{\vec{m}_i}(p_i t)) e^{-\vec{p}_i t})\right) dt \tag{2}$$

Likewise, we have the special case where all coupons are equally likely and $\vec{m}$ is a constant.

Corollary 1 (Coupon Collecting with Uniform Probability [19])**.**

$$T_k(m, 1) = k \int_0^\infty \left(1 - \prod_{i=1}^{k} (1 - S_{m_i}(t) e^{-t})\right) dt = \tag{3}$$

$$k(\log(k) + (m-1)\log(\log(k)) + o(1)) \tag{4}$$

Using Theorem 2 and Corollary 1 we can then construct the following theorem from [22].

Theorem 3 (Coupon Collecting Upper Bound [22]). *Let p_{max} and p_{min} be the maximum and minimum values of $\vec{p}$ respectively. Let m_{max} be the maximum value of $\vec{m}$. Then we have the following*

$$T_k(\vec{m}, \vec{p}) \leq \frac{p_{max}}{p_{min}} T_k(m_{max}, 1) = \tag{5}$$

$$\frac{p_{max}}{p_{min}} k(\log(k) + (m_{max} - 1)\log(\log(k)) + o(1)) \tag{6}$$

3 PROBLEM FORMULATION

3.1 Setting

We consider the same setting as [20] adapted for online query answering.

There are k analysts each with an associated positive privacy budget $s_1, s_2 \ldots s_k$ such that $\sum_{i=0}^{k} s_i = \epsilon$. These weights denote the shares of the overall privacy budget ϵ to which each analyst is entitled. We denote the privacy budget entitled to a collection of analysts $\mathcal{S} \subseteq \{1, ..., k\}$ as $\epsilon_{\mathcal{S}} = \sum_{i \in \mathcal{S}} s_i$.

A Differentially Private Multi-Analyst online mechanism takes in as input a sequence of query, analyst tuples $(q, i) \in Q \times \{1, \ldots, k\}$ consisting of a linear counting query and the identifier for the analyst asking the query. We denote the sub-sequence of Q containing only the queries asked by analyst i as Q^i and the sub-sequence of Q containing queries from analysts in the set $\mathcal{S}$ as $Q^{\mathcal{S}}$. Likewise we denote the ith query in the query sequence Q_i. The data curator must answer each query one at a time before the next query is revealed.

Bun et al. [1] show that answering queries in the online setting is strictly more difficult than in the offline setting since the queries are not known in advance. This leads to classes of queries that can be answered efficiently under differential privacy in the offline case but not in the online case. Mechanisms designed for the offline case do not always translate straightforwardly to the online case. We will show that the additional constraints of the sharing incentive causes an even greater separation between the offline and online multi-analyst settings. There are additional complexities that arise when considering multiple analysts in the online setting. In particular, we will show that the order in which queries are received can affect the distribution of error across queries. This is a unique property of the online setting which makes it difficult for a mechanism to share privacy across analysts budget while still ensuring the sharing incentive.

3.2 Desiderata

The utility metrics used in online QA systems (number of sufficiently accurately answered queries) are often different from those used in offline QA systems (total mean squared error). In this work, we define utility as the number of queries an analyst can answer with error under some pre-defined threshold α. Of course, the correct notion of utility may be context dependent, so we define the desiderata with respect to an arbitrary utility function U_i representing the utility for analyst i. Given this, we adapt the multi-analyst desiderata from [20] to the online setting. At a high level, these desiderata are intended to ensure that each of the multiple analysts receives high utility, as opposed to merely optimizing for total or average utility, potentially at the expense of some analysts seeing poor performance.

A natural baseline for the utility a given analyst should demand is that which they would expect if their queries were answered independently with their share of the privacy budget. Any multi-analyst differentially private mechanism should incentivize a rational agent to participate in the collective mechanism by guaranteeing that they will receive at least as much utility as they would expect in the independent case. Any mechanism which satisfies this requirement is said to satisfy the *sharing incentive*, more formally as follows.

Definition 3 (Sharing Incentive [20]). *A mechanism $\mathcal{M}$ satisfies the sharing incentive if for any collection of analysts $\mathcal{S}$, any analyst $i \in S$ and all query sequences,*

$$\mathbb{E}[U_i(\mathcal{M}, Q^{\mathcal{S}}, \epsilon_{\mathcal{S}})] \geq \mathbb{E}[U_i(\mathcal{M}, Q^i, s_i)]$$

Where $\mathbb{E}$ is the expectation over the randomness of the mechanism. The sharing incentive only provides a baseline of comparison to utility in the independent case. A stronger guarantee would be that the addition of an analyst (along with their privacy budget) to any collective should never decrease the utility of any analyst in the collective. This property is called *analyst monotonicity.*

Definition 4 (Analyst Monotonicity [20]). *A mechanism $\mathcal{M}$ is analyst monotonic if for all collections of analysts $\mathcal{S}$, any two analysts $i, j \in \mathcal{S}$ for all query sequences.*

$$\mathbb{E}[U_i(\mathcal{M}, Q^{\mathcal{S}}, \epsilon_{\mathcal{S}})] \geq \mathbb{E}[U_i(\mathcal{M}, Q^{\mathcal{S}\setminus j}, \epsilon_{\mathcal{S}\setminus j})]$$

It is straightforward to show that analyst monotonicity implies sharing incentive by induction. The converse, however, is not true; analyst monotonicity is a stronger guarantee in this sense. For example, suppose you know the utilities guaranteed by the sharing incentive to analysts from their independent cases. You could satisfy the sharing incentive by optimizing for total number of queries answered subject to utility constraints for all of the analysts, but such a scheme would not ensure analyst monotonicity in general.

Note that checking for violation of analyst monotonicity empirically is intractable in general, as it requires quantifying over all possible subsets of analysts. The relaxation of analyst monotonicity called *non-interference* provides the same guarantee but only for the collective consisting of all analysts. In other words, the weaker guarantee of non-interference is that the addition of the *last* analyst to the collective does not decrease the utility of any other analyst. We will argue theoretically that our algorithms satisfy the stronger property of analyst monotonicity or the weaker property of sharing incentive, but we include the definition of non-interference because we measure the empirical interference in our experiments.

Definition 5 (Non-Interference [20]). *A mechanism $\mathcal{M}$ satisfies non-interference if for any two analysts $i, j \in \mathcal{S}$ where $i \neq j$, and for all query sequences,*

$$\mathbb{E}[U_i(\mathcal{M}, Q, \epsilon)] \geq \mathbb{E}[U_i(\mathcal{M}, Q \setminus Q^j, \epsilon - s_j)]$$

Finally, any multi-analyst mechanism should be able to adapt and efficiently answer any valid query. We say that a multi-analyst mechanism is *Adaptive* if the sequence of outputs given is a direct function of its input queries and is not pre-determined. This ensures that in addition to satisfying the desiderata above mechanisms are efficient and non-trivial.

3.3 Independent Mechanisms

Prior work [20] has shown that one can satisfy all three desiderata by dividing the privacy budget and running an instance of a single analyst differentially private mechanism (with their associated privacy budget). These *independent mechanisms* however are highly inefficient as they do not allow analysts with similar queries to share either privacy budget or query answers. We therefore, take these independent mechanisms as an appropriate baseline to which we can compare our new mechanisms.

3.4 Problem Statement

The goal of this work is to design online multi-analyst differentially private mechanisms that answer (possibly large) query sequences submitted by multiple analysts while satisfying the three desiderata.

Problem 1. *Given any sequence of query, analyst tuples Q on a database D with positive weights $s_1, \ldots, s_k$ s.t. $s_1 + \ldots + s_k = \epsilon$, design an adaptive mechanism $\mathcal{M}$ such that:*

- *$\mathcal{M}$ satisfies differential privacy,*
- *$\mathcal{M}$ satisfies sharing incentive (Definition 3), analyst monotonicity (Definition 4), and non-interference (Definition 5),*
- *and $\mathcal{M}$ answers as many queries accurately as possible.*

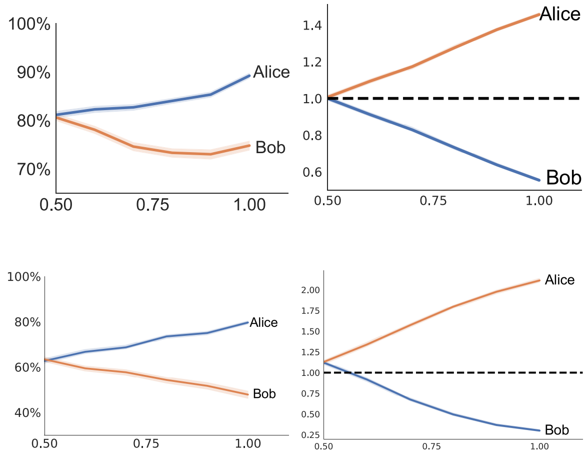
4 MOTIVATING EXPERIMENTS

Here we demonstrate that both classic mechanisms such as the Laplace mechanism as well as state of the art mechanisms such as Private Multiplicative Weights fail to satisfy the sharing incentive and non-interference. The experiments highlight a key difference between the online and offline problem, namely that the order in which queries are answered can have a significant impact on the utility of the agents.

Consider the case with two analysts, Alice and Bob, each with an identical query sequence on disjoint halves of the dataset and an equal share of half the privacy budget $\epsilon = 1$. In order to generate the query sequences, we let $p \in [0.5, 1]$ determine the probability that Alice's query gets asked at each time step. At $p = 0.5$, Alice and Bob have an equal chance of their queries being answered; the resulting joint query stream is uniformly distributed between their queries. As p increases at each time step the probability that the query will belong to Alice increases. When $p \rightarrow 1$, all of Alice's queries are answered first followed by Bob's queries.

In the following experiments, the online mechanisms used were the Laplace mechanism and Private Multiplicative Weights (PMW) [11]. We use the practical census database provided in [18]. This database contains information about population migration by age and is of dimension $|x| = 86$ representing ages from 0 to 85.

For the following figures, we generated a randomized workload, containing point and range queries, 1000 times, and ran PMW for each value of p, in $[0.5, 0.6, 0.7, 0.8, 0.9, 1]$. We examined the percent of queries answered with error under the threshold $\alpha = 0.01$ for each analyst's workload (Fig 1a) and the Ratio Error (Fig 1b), a metric first introduced in [20]. The ratio error measures the ratio between the number of queries answered with error under α in the joint case and the number of queries answered with error under α in the independent case for each analyst. Values above 1 signify a violation of the sharing incentive with larger values signifying a larger violation. The shaded regions of the plot represent the 90% confidence intervals.



(a) Queries Answered **(b) Max Ratio Error**

Figure 1: Empirical Measures for PMW (above) and Laplace (below). Values of Utility (Left) and Sharing Penalty (Right) are shown with varying values of p (x-axis).

The following results are consistent across both mechanisms. In Figure 1a, we show that utility disparities between analysts grow as p increases. Alice and Bob have the same utility at $p = 0.5$ but as p increases, Alice's utility increases and Bob's utility decreases. Even in a situation with only 2 analysts with very similar queries, the ordering of the queries can significantly impact the utility of each analyst. In a system designed without multi-analyst desiderata in mind, those analysts who ask their queries early may receive a large benefit for doing so. This could create an incentive for analysts in such systems to "race" to submit queries first to ensure higher utility.

Figure 1b, demonstrates that this increased disparity can cause a mechanism to fail to satisfy the sharing incentive. Here we test an instance where Alice and Bob have identical query sequences on disjoint partitions of the database (e.g., males and females). Although they start with the same max ratio error when p = 0.5, the disparity grows as p grows. This increased disparity causes a failure to satisfy the sharing incentive as, in the worst case, Bob receives 40% more error than he would have in the independent case.

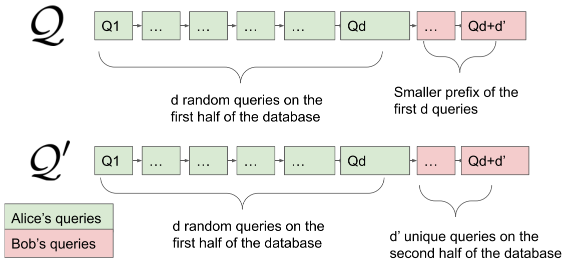

$\mathcal{Q}$
Q1
...
...
...
...
Qd
...
Qd+d'
d random queries on the first half of the database
Smaller prefix of the first d queries
$\mathcal{Q}'$
Q1
...
...
...
...
Qd
...
Qd+d'
Alice's queries
Bob's queries
d random queries on the first half of the database
d' unique queries on the second half of the database

Figure 2: Visualization of the two sequences Q and Q'

5 LIMITATIONS IN THE ONLINE SETTING

Here we will introduce a fundamental performance cost for online mechanisms satisfying the sharing incentive. Specifically, we prove Theorem 6, which shows that there is an upper bound to the number of queries that can be accurately answered while satisfying the sharing incentive, and that the number of queries decreases with k, the number of analysts.

The argument takes advantage of the online nature of the mechanism. We provide two sequences Q and Q' which contain the same (significantly long) prefix. We will argue that in order to satisfy the sharing incentive and answer all the queries with expected error below an arbitrary threshold α, the mechanism would need to have different behavior during the prefix in both Q and Q'. Since the prefixes are identical no online mechanism can differentiate between the two. Consequently, a mechanism must choose between answering many queries consistently and satisfying the sharing incentive.

In order to establish the result, we rely on a few previous theorems. First, we require the lower bound on error for any pure $(\epsilon, 0)$-differentially private mechanism.

Theorem 4 (Differential Privacy Lower Bound [12]). *Any mechanism M which satisfies differential privacy must suffer at least $\Omega(|Q|/\epsilon) \cdot \min\left(\sqrt{\log(n/|Q|)}, \sqrt{|Q|}\right)$ expected error*

We then need to establish the existence of a mechanism which can answer a large enough number of queries with a set privacy budget.

Theorem 5. *There exists an ϵ differentially private query answering mechanism which can answer $|Q|$ queries, each with error less than or equal to $\frac{|Q|\sqrt{|Q|}}{\epsilon}$*

Proof. By splitting the privacy budget evenly across each of the $|Q|$ queries then applying the Laplace Mechanism to answer each query you can answer $|Q|$ queries with expected error exactly $\frac{|Q|\sqrt{|Q|}}{\epsilon}$ [5]. By Theorem 1 this satisfies ϵ-differential privacy. □

Theorem 6 (Sharing Incentive Query Limit). *For all online multi-analyst ϵ-DP query answering mechanisms M, for any number* *of analysts k, any database x of dimension $|x| \geq 2k$ and n sufficiently large there exists shares of privacy budgets $(s_1, \ldots, s_k)$ and a query sequence Q of size $O((\alpha\epsilon/k)^{2/3})$ such that if mechanism $\mathcal{M}$ can answer all the queries in Q with error less than threshold α then there exists an alternative workload Q' in which $\mathcal{M}$ must violate the sharing incentive.*

Proof. We begin by considering the case where there are two analysts, Alice and Bob and will expand to k analysts afterward. Each analyst is entitled to $\epsilon/2$ of the privacy budget. We will construct two query sequences Q and Q' which share a large prefix such that if a mechanism spends the entire privacy budget answering the prefix it can answer all the queries in Q but if it spends more than $\epsilon/2$ budget answering the prefix, it will violate the sharing incentive in Q'.

We start by constructing sequence Q. First we partition the database on a single predicate into two disjoint halves (e.g., males and females). Alice will ask queries on the first half of the database. Bob will also ask his queries on the first half in Q. He will ask his queries on the second half in Q'. Alice asks d random queries on the first half of the database where d is the largest integer such that $\frac{d\sqrt{d}}{\epsilon} < \alpha < \frac{2d\sqrt{d}}{\epsilon}$. By Theorem 4 Alice's queries cannot be answered by any mechanism using only $\frac{\epsilon}{2}$ of the privacy budget but can with the entire privacy budget. Bob will then ask the first d' queries from Alice's queries where d' is the largest integer such that $\frac{2d'\sqrt{d'}}{\epsilon} < \alpha$. In this case, all of Bob's queries are identical to Alice's queries and as such her query answers can be reused to answer Bob's.

Now we will construct the alternative sequence Q'. Alice will ask the same queries as in Q, this will serve as the identical prefix. In this case, Bob will ask a different set of queries. Bob will ask d' random distinct queries from the second half of the database where d' is the largest integer such that $\frac{2d'\sqrt{d'}}{\epsilon} < \alpha < \frac{(d+d')\sqrt{(d+d')}}{\epsilon}$. In this case, since Bob's queries are from the second half of the database Alice's queries cannot be used to aid in answering Bob's queries.

Since Bob's queries in Q are copies of Alice's queries, all the queries in Q can be answered by answering only Alice's queries using the entire privacy budget and reusing her answers to answer Bob's queries. By Theorem 5 the Laplace mechanism can answer all of Alice's queries under the threshold α. Since $\alpha < \frac{2d\sqrt{d}}{\epsilon}$, by Theorem 4, there is no differentially private mechanism that can answer Alice's queries while only using $\epsilon/2$ of the privacy budget. In order to answer all of Alice's queries some of Bob's budget must be used. Likewise in Q' since $\frac{(d+d')\sqrt{(d+d')}}{\epsilon} > \alpha$ there exists no ϵ-differentially private mechanism that can answer all of Alice's queries and all of Bob's queries under the threshold. However, no online mechanism can distinguish between Q and Q' before Bob's queries and thus must have the same behavior on both sequences up to this point. Since Alice's queries in both are sufficiently large the mechanism must decide on either answering all of Alice's queries or saving privacy budget for Bob. If mechanism $\mathcal{M}$ can answer all the queries in Q it must use Bob's share of the privacy budget prior to Bob's queries being answered. In Q this is fine as Bob's queries consist of a prefix of Alice's queries which can be reused, however in Q', since d' is the largest integer such that $\frac{2d'\sqrt{d'}}{\epsilon} < \alpha$ by Theorem 4, if any of Bob's privacy budget is used prior to his queries there exists no mechanism that can answer all of his queries. Since Bob's share of the privacy budget is sufficient for mechanism $\mathcal{M}$ to answer all of his queries in the independent case, this is a violation of the sharing incentive. Alternatively, if mechanism $\mathcal{M}$ does satisfy the sharing incentive it cannot answer all of Alice's queries in either Q or Q' since it cannot distinguish between Q and Q' prior to Bob's queries.

We can extend this example to the case of k analysts to get the linear separation between a general online mechanism and one that satisfies the sharing incentive. We consider k analysts each with $\frac{\epsilon}{k}$ of the privacy budget and split the database into k equal parts. The first analyst will ask d queries where d is the largest integer such that $\frac{d\sqrt{d}}{\epsilon} < \alpha < \frac{kd\sqrt{d}}{\epsilon}$.

Each subsequent analyst will each ask a set of d' queries where d' is the largest integer such that $\frac{kd'\sqrt{d'}}{\epsilon} < \alpha$ and $\frac{(d+(k-1)d')\sqrt{(d+(k-1)d')}}{\epsilon} > \alpha$. In Q each of those analysts will ask prefixes of the first analyst's queries and in Q' they will each ask queries from their unique partition of the database.

By Theorem 4 no mechanism can answer more than$(\alpha\epsilon/k)^{2/3}$ queries using with error under α using $\frac{\epsilon}{k}$ of the privacy budget. As such if the mechanism answers any more than $(\alpha\epsilon/k)^{2/3}$ of the first analyst's queries it must take from at least one of the other subsequent analysts resulting in a violation of the sharing incentive.

Therefore any mechanism that satisfies the sharing incentive in all cases can answer at most $(\alpha\epsilon/k)^{2/3}$ queries for any agent when the privacy budget is distributed equally. Now consider Q'' which shares the same prefix as Q and Q'. In this case, each analyst other than the first only asks one query. Since this query sequence contains the same large prefix it must have the same behavior as in the previous two cases. Thus despite being asked many queries a mechanism that satisfies the sharing incentive can only answer $(\alpha\epsilon/k)^{2/3} + k - 1$ queries in total. □

This shows that any online mechanism can choose between answering a large number of queries in all cases or satisfying the sharing incentive but not both. Compare this to the number of queries that can be answered by the simple Laplace mechanism with sequential composition. The Laplace mechanism can answer an arbitrary $(\alpha\epsilon)^{2/3}$ queries under the threshold α. This means that even the Laplace mechanism with sequential composition can answer too many queries, in any sequence of queries, to satisfy the sharing incentive. Any mechanism which satisfies the sharing incentive cannot perform any better in all cases than the independent version of the Laplace mechanism which divides the privacy budget equally k ways.

We note that this result is unique to the online setting where queries must be answered one at a time and in a fixed order. In the offline setting where the entire workload is known one can create an alternative "strategy workload" to answer which can be used to reconstruct each analysts' queries efficiently. Often online mechanisms are analyzed through the random order model [7–9] where the queries appear in random order instead of an adversarial one. However, in our case ordering the queries randomly doesn't circumvent the upper bound of Theorem 6 since, in the setting above, the first analyst asks the vast majority of queries. As the number of analysts and privacy budget grows large the first analyst's queries become the overwhelming majority of queries and as such the adversarial case will still happen with high probability.

In the following sections, we will present two solutions for online multi-analyst differential privacy. The first solution in Section 6 will be an analyst monotonic mechanism for any sequence of queries and any distribution of privacy budget. As such this solution will be subject to the upper bound of Theorem 6. In Section 7 we will introduce a method to circumvent Theorem 6 by restricting the order in which queries can appear. From there we will show that any sufficiently efficient online mechanism can be made to satisfy the sharing incentive by restricting the order in which analyst queries are answered.

6 CACHE AND RECONSTRUCT

Here we introduce Seeded Cache and Reconstruct (Algorithm 1), an online mechanism for answering linear queries which satisfies all the desiderata for any sequence of queries in any order. The seeded cache and reconstruct mechanism works by initially generating a cache of answered queries and then using that cache to reconstruct other queries once privacy budget has been expended. The mechanism has three phases which each analyst goes through asynchronously based on their remaining privacy budget:

- The first phase happens during initialization. Each analyst donates a fixed fraction of their privacy budget to answer a basis set of queries using all the donated budget. As the number of analysts increases the quality of the seed is increased as more privacy budget is donated to it. An example of such a basis is a histogram of counts over all unique values in the database's universe.
- The second phase begins after initialization and ends once the analyst runs out of privacy budget. In this phase, the analysts ask additional queries. If the query is already in the cache the answer is reused. If the query is not in the cache it is answered using the Laplace mechanism and then added to the cache.
- The third phase begins once an analyst's privacy budget is expended. In this phase, they can use the entire cache to reconstruct their remaining queries by using the Matrix Mechanism [15] reconstruction step. Since the basis was generated in the first phase it is possible to reconstruct any query from the cached queries but those reconstructions may lead to answers with high error.

Lemma 3. *Seeded Cache and Reconstruct is analyst monotonic.*

Proof. Consider a collective $S \subseteq \{1, ..., k\}$ and analysts $i, j \in S$. $Q^{\mathcal{S}\setminus j}$ is the subsequence of Q restricted to queries labeled for analysts in $\mathcal{S} \setminus j$. Consider some $q \in Q^i$ such that Seeded Cache and Reconstruct answers q with error at most α when run on $Q^{\mathcal{S}\setminus j}$ with privacy budget $\epsilon_{\mathcal{S}\setminus j}$. We will show that Seeded Cache and Reconstruct also answers q with error at most α when run on $Q^{\mathcal{S}}$ with privacy budget $\epsilon_{\mathcal{S}}$. The argument follows by cases for how q is answered by Seeded Cache and Reconstruct in the first instance (without j in the collective).

First, suppose q was answered directly by using privacy budget from analyst i in the first instance. Then q will still be answered as long as i has sufficient privacy budget when asking q in the second instance. This must be the case: For any prior query on which i spends privacy budget in the first instance, either i still spends the same privacy budget on the query in the second instance, or the query was asked by j and cached, in which case i will have more privacy budget in the second instance. In other words, individual analyst privacy budgets at a given query in their sequence are monotone non-decreasing with respect to the addition of an analyst to the collective.

Second, suppose the answer to q was in cache in the first instance: Then some other analyst $k \neq i \in \mathcal{S} \setminus j$ spent budget to answer q prior to i's query in $Q^{\mathcal{S}\setminus j}$. Because answering directly always takes precedence over reconstruction when possible, k will also spend budget to answer q prior to i's query in $Q^{\mathcal{S}}$ unless j has already answered and cached q previously. Either way, q will still be cached when i asks the query in $Q^{\mathcal{S}}$. Note that k must still have enough budget to afford q in $Q^{\mathcal{S}}$ by the budget monotonicity condition argued in the first case. In other words, the cache contents are also monotone with respect to adding an analyst to the collective.

Third and finally, suppose q was answered by reconstruction using the Matrix Mechanism reconstruction step in the first instance (with expected error at most α). Suppose for a contradiction Seeded Cache and Reconstruct returned a reconstructed answer to q with expected error greater than α in the second instance with j in the collective. But we have already argued in the second case that the cache contents are monotone with respect to adding an analyst to the collective, and Lemma 2 states that any such reconstruction on a superset of the first instance's cache can only have lower expected error. Likewise, the seed quality monotonically increases with the number of analysts. Lemma 1 states that any reconstruction with identical queries of higher quality can only have lower expected error. This establishes the contradiction in the third case.

Thus, the queries of analyst i answered in the first instance (without j in the collective) with expected error at most α is a subset of the queries answered with expected error at most α in the second instance (with j in the collective). As the analyst's utility is the number of such queries, it follows that i's utility in the first instance is no more than i's utility with j in the collective.

□

6.1 Utility and Trade-offs

Each component to seeded cache and reconstruct adds additional utility to the mechanism while retaining analyst monotonicity. The reconstruction allows for additional queries to be answered once an analyst's privacy budget is expended. The full rank seeding at the beginning ensures that the matrix mechanism reconstruct step [15] can reconstruct any linear query and not just those supported by the queries that are added to the cache. This ensures that even when all analysts have expended their privacy budget any query may be answered (though not necessarily with error below the α threshold).

Seeded cache and reconstruct always satisfies Analyst Monotonicity, regardless of the seed and parameter choice, (and thus the weaker notion of Sharing Incentive) but the choice of seed can

```
Algorithm 1: Seeded Cache and Reconstruct
input  : Sequence of queries with associated analyst
         $Q \leftarrow (q_1, a_i) \ldots (q_{|Q|}, a_i)$,
         Vector of $k$ shares $\vec{S} \leftarrow \{s_1, s_2 \ldots s_k\}$ ,
         Data vector $\mathbf{x}$,
         privacy budget $\epsilon$,
         Threshold $\alpha$,
         Fraction of budget for seed $\gamma$,
         Basis of queries to be generated $\boldsymbol{B}$,
         Privacy budget per query $\lambda$
output : Sequence of Query Answers

Phase 1 (Mechanism Initialization)
Seed cache $C$ with all queries of $B$ with $\gamma\epsilon$ privacy budget
Create a vector of remaining privacy budget $\vec{\epsilon} = (1-\gamma)\epsilon\vec{S}$
for $i \leftarrow 1$ to $|Q|$ by $1$ do

    Phase 2 (Using Privacy Budget)
    if $\vec{\epsilon}_i \geq \lambda$ then
        if $Q_i \in C$ then
            return Query answer from $C$
        else
            Create noisy query answer $\hat{Q}_i$ with Laplace
              Mechanism using privacy budget $\lambda$
            $\vec{\epsilon}_i \leftarrow \vec{\epsilon}_i - \lambda$
            Add $\hat{Q}_i$ to $C$
            return $\hat{Q}_i$

    Phase 3 (Reconstructing Queries)
    if $\vec{\epsilon}_i < \lambda$ then
        return Matrix Mechanism reconstruction of $Q_i$
          using $C$
```

greatly impact the error of reconstructed queries. Seeded cache and reconstruct is at its best when one has some prior knowledge of the likely kinds of queries to be asked. In this case, the data curator can choose a specific seed that performs well on those queries. Past work [2, 13, 21, 24] has shown that one can design a particular workload that performs well on a family of queries. For example the hierarchical mechanism [21] performs especially well on long range queries. A data curator who knows that range queries will be asked frequently may use the queries from the hierarchical mechanism as a basis in that case. Even in cases where there is no known optimal basis for a particular family of queries the data curator can use the Matrix Mechanism [15] to generate a strong basis.

One may note that seeded cache and reconstruct is restrictive in how it answers queries: the algorithm never attempts to reconstruct the answer to a new query if there is privacy budget available to answer it via Laplace. There are times when this seems to imply an unnecessary waste of privacy budget. We argue that this restriction is necessary to ensure analyst monotonicity, specifically the sharing incentive.

For example, consider the case where there are two analysts Alice and Bob. They each have equal shares of the privacy budget. Assume that any query reconstructed with at most 2 queries which are answered using the privacy budget will have error under the threshold α. Alice asks her queries first and asks all the point queries (range queries of length 1). Bob asks all the length 2 range queries followed by all the length 4 range queries. In the independent case Bob will answer all of his length 2 range queries by expending his privacy budget and will reconstruct his length 4 range queries using those answers. In the joint case however Alice will answer all her queries using her privacy budget, then Bob will reconstruct the size 2 range queries from Alice's cached queries. When he asks the size 4 range queries he will not be able to reconstruct them all from the queries in the cache and will fail in answering all of them with privacy budget directly. These conditions arise when the addition of new analysts results in drastically different behavior between the joint and independent cases.

Since seeded cache and reconstruct satisfies the sharing incentive it is subject to Theorem 6. Seeded cache and reconstruct performs particularly poorly in cases like the ones shown in Section 5 where each analyst asks queries on disjoint sections of the database. In these cases, no two analysts have any queries that rely on the same information. As no analyst can use another analyst's query answers to help reconstruct their own. In this case, each analyst performs no better than if they had answered their queries independently. While these cases do exist we show in Section 8 that these cases are infrequent and that cache and reconstruct typically outperforms even optimal independent mechanisms.

Despite this, there is a cost that analyst monotonic mechanisms pay in terms of their ability to a large number of queries below α error. We see in Section 8 that mechanisms that fail to satisfy any of the desiderata such as non-independent PMW outperform seeded cache and reconstruct in this sense.

7 THE QUERY SCHEDULER

Seeded Cache and Reconstruct achieves all of our desiderata, and in a sense "solves" the problem of multi-analyst differential privacy for online query answering. However, it is subject to the fundamental upper bound of Theorem 6 which implies that performance (measured in the total number of queries answered) has a polynomial dependence on k, the number of analysts. In system design contexts where privacy budget is an extremely precious resource, this may not be an acceptable trade-off.

Instead, one might want to use a given state-of-the-art online query answering algorithm without compromising performance and simply modify it to satisfy the sharing incentive for multiple analysts. In this section, we describe how to complete such a generic reduction. Our intuition from Section 4 is that the *order* in which analysts ask queries is also of crucial importance to multi-analyst desiderata (in particular, it is better from an analyst's perspective to come earlier in the order). Whereas Seeded Cache and Reconstruct achieves analyst monotonicity by directly accounting for the privacy budget used by different analysts, we show that it is also possible to satisfy the sharing incentive by constraining the order in which analysts ask queries.

We introduce the **Query Scheduler**. The query scheduler takes in as input an existing online mechanism and the query sequence. At each time step, an analyst is chosen to answer their next query in the sequence. While seeded cache and reconstruct allocates privacy budget directly, the Query Scheduler instead "allocates" *time*, by selecting at which times analysts can answer their queries. Note that this approach circumvents Theorem 6 precisely by restricting the online input model. Importantly, we do not constrain the set of possible query sequences, only the set of possible analyst identifier sequences. Practically speaking, this has the effect of modifying existing algorithms to satisfy the sharing incentive at the cost of occasionally "stalling" when the next scheduled analyst does not have a query ready.

Below we introduce two methods for allocating time across analysts. The **round-robin Scheduler** enforces that queries be asked in a round-robin fashion where every analyst must answer a query before any other analyst can ask another. The **Randomized Scheduler** instead randomly selects an analyst at each time step to answer their queries.

7.1 Round-robin Scheduler

The round-robin scheduler selects analysts one at a time in a fixed rotation. This ensures that every analyst is given a chance to answer a query before any analyst gets another. Consider applying the round-robin scheduler to the adversarial query sequence from Section 5 where Alice asks all her queries first followed by a single query from each other analyst. The round-robin scheduler will ensure that each analyst receives an opportunity to ask their queries before Alice expends all of the privacy budget. As a result the scheduler stalls after Alice's first query waiting for the queries from the other analysts. In this case, Alice must wait until the entire query sequence is complete before her second query is answered. The round-robin scheduler ensures that each analyst will have the opportunity to answer their queries before all the resources are consumed at the cost of additional wait time. As a result Theorem 7 states that any mechanism which scales at least linearly with the privacy budget satisfies the sharing incentive.

Theorem 7 (round-robin). *Let every analyst have an equal share of the privacy budget. When queries are asked in a round-robin manner any mechanism $\mathcal{M}$ which can answer at most c queries with expected error under threshold α with privacy budget ϵ and can answer at least the first $k \cdot c$ queries under threshold α with privacy budget $k \cdot \epsilon$ satisfies the sharing incentive.*

Proof. First, we note that as a property of the round-robin scheduler if there are k analysts after answering $c \cdot k$ queries each analyst has asked exactly c queries. Each analyst is entitled to an equal privacy budget $\frac{\epsilon}{k}$. Therefore when there are k analysts participating there is a total of ϵ privacy budget. By assumption mechanism $\mathcal{M}$ can answer at most c queries and at the end of c queries each analyst has had $\frac{c}{k}$ queries answered. Let an additional analyst join the system. The total privacy budget is now $\epsilon + \frac{\epsilon}{k} = (1 + \frac{1}{k}) \cdot \epsilon$. By assumption mechanism $\mathcal{M}$ can answer at least $c + \frac{c}{k} = \frac{(k+1)c}{k}$ queries resulting in each analyst answering at least $\frac{(k+1)c}{k} \frac{1}{k+1} = \frac{c}{k}$ queries. Therefore the addition of any analyst can only ever improve the number of queries each analyst answers. □

We can further adapt this for cases with non-equal shares of the privacy budget. Instead of doing strict round-robin ordering during each round we allow an analyst to ask a number of queries proportional to the number of queries they could ask in the independent setting.

Unfortunately, the round-robin scheduler depends on each analyst always having a query ready when it is their turn to answer. In order to preserve the sharing incentive if an analyst does not have a query ready the Scheduler must stall and wait for their query. In practical settings, this could lead to long times waiting while the mechanism is stalled.

7.2 Randomized Scheduler

The randomized scheduler, like the round-robin scheduler, selects an analyst at each time step to answer a query. Unlike the round-robin scheduler which answers queries in a deterministic order, the randomized scheduler randomly selects an analyst at each time step. The randomized scheduler is less restrictive than the round-robin scheduler while still ensuring that an efficient enough mechanism can satisfy the sharing incentive without any further changes. This can be demonstrated through a reduction to the well studied coupon collectors problem [19, 22]. In the coupon collectors problem, there are k unique types of coupons in an urn. At each time step one coupon is chosen at random (with replacement). The goal is to find the value of $T_k(\vec{m}, \vec{p})$, that is the expected number of time steps required to acquire $\vec{m}_i$ of each coupon when there are $\vec{p}_i$ of each coupon in the urn.

We can ask a similar question of the randomized scheduler. At each time step, given that an analyst is chosen at random to answer their query, in expectation how many queries must be answered so that each analyst answers at least as many queries as in the independent case. This reduces directly to the coupon collectors problem if you set $\vec{m}$ to be the number of queries that each analyst receives in the independent case $T_k(\vec{m}, \vec{p})$ becomes the expected number of queries required for each analyst to at least answer as many queries as in the independent case. As such we can prove Theorem 8.

Theorem 8 (Uniform Randomized Scheduler). *Let each of the k analysts have an equal share of the privacy budget. Assume that at each time step an analyst is selected uniformly at random to answer a query. Given a mechanism $\mathcal{M}$ which can answer at most c queries with expected error under threshold α with privacy budget ϵ and can answer at least the first $k(\log(k) + (c-1)\log(\log(k)) + o(1))$ queries under threshold α with privacy budget $k \cdot \epsilon$ satisfies the sharing incentive.*

Proof. If a random analyst is chosen at each round to answer a query by Corollary 1 the expected number of queries required to satisfy the sharing incentive for all analysts is $T_k(c, 1) = k(\log(k) + (m-1)\log(\log(k)) + o(1))$. □

We can use Theorem 2 to extend this statement to hold for any distribution of privacy budgets and non-uniform distribution.

Theorem 9 (Non-Uniform Randomized Scheduler). *Let $\vec{p}$ be the vector with values proportional to the probabilities that each analyst is chosen during any time step and $\vec{m}$ be the vector that states the number of queries each analyst can answer independently. Given a mechanism $\mathcal{M}$ which can answer at most $\vec{m}_i$ queries with expected* *error under threshold α with privacy budget $s_i\epsilon$ and can answer at least the first $\frac{p_{max}}{p_{min}}k(\log(k)+(m_{max}-1)\log(\log(k))+o(1))$ queries under threshold α with privacy budget ϵ satisfies the sharing incentive, where p_{max} and p_{min} are the maximum and minimum values of $\vec{x}$ respectively and m_{max} is the maximum of $\vec{m}$.*

Proof. First we note that Theorem 3 upper bounds $T_k(\vec{m},\vec{p}) \leq \frac{p_{max}}{p_{min}}T_k(m_{max},1)$. We then once again apply Corollary 1 directly to get $\frac{p_{max}}{p_{min}}k(\log(k)+(m_{max}-1)\log(\log(n))+o(1))$ thus proving Theorem 9. □

We note that while Theorem 9 implies that any mechanism efficient enough to answer that many queries will always satisfy the sharing incentive it is not always the case that such a mechanism exists. For instance, if $T_k(\vec{m},\vec{x})$ is large enough then the lower bounds of [3, 12] ensure that no mechanism can answer all the queries under threshold α.

Algorithm 2: Randomized Scheduler

```
input  : Sequence of queries with associated analyst
         $Q \leftarrow \{(q_1, a_i)\dots(q_{|Q|}, a_i)\}$,
         Number of analysts $k$
         Differentially private mechanism $\mathcal{M}$,
         Set of parameters $\mathcal{P}$,
output : Sequence of Query Answers

Initialize mechanism $\mathcal{M}$ with parameters $\mathcal{P}$
Create a buffer $\vec{B}$ of queries for each analyst

At each time step $i$
if $i \leq |Q|$ then
    Add $Q_i$ to the buffer of the associated analyst
Sample $j \in [1, k]$ uniformly at random
return The first query in analyst $j$s buffer with $\mathcal{M}$
```

7.3 Utility and Trade-offs

Unlike seeded cache and reconstruct the query schedulers enforce the sharing incentive while circumventing the upper bound of Theorem 6. The query scheduler inherits the efficiency guarantee of the online mechanism that it uses. This ensures that the query scheduler can answer just as many queries as a traditional online mechanism that doesn't satisfy any of the desiderata and can answer significantly more queries than independent mechanisms.

In the standard online model mechanisms that satisfy the sharing incentive incur a penalty to the number of queries they can answer. The query scheduler instead incurs a penalty to how fast it can answer those queries. Since the query scheduler enforces which analysts can ask queries at any given time it cannot progress if the chosen analyst has no queries to be answered. This causes the query scheduler to stall and wait for an analyst to ask more queries (or indicate that they are done asking queries), preventing any other analyst from asking their queries. We measure the impact of stalling in practice by measuring each mechanism's time to completion in Section 8.

The round-robin scheduler ensures that any efficient mechanism satisfies the sharing incentive however severely restricts the ordering of analysts. By ensuring that all analysts have asked the same amount of queries the round-robin scheduler is guaranteed to stall if any analyst has fewer queries prepared than any of the others. The randomized scheduler requires an underlying mechanism that is more efficient than the round-robin scheduler. In exchange, the randomized scheduler stalls less often as the order of analysts is sampled from a distribution which can be chosen to match some prior knowledge of analysts' behavior. We show in Section 8 that this is particularly desirable when some analysts ask significantly fewer queries than others. In that case, even when the analyst chosen is sampled from the uniform distribution it incurs less stalling time than the round-robin scheduler.

8 EXPERIMENTS

We design experiments to both test if the mechanisms proposed satisfy the desiderata as well as how well they perform in practice. The randomized process used to generate sequences is designed to emulate possible sequences across multiple analysts that one might see in a practical setting.

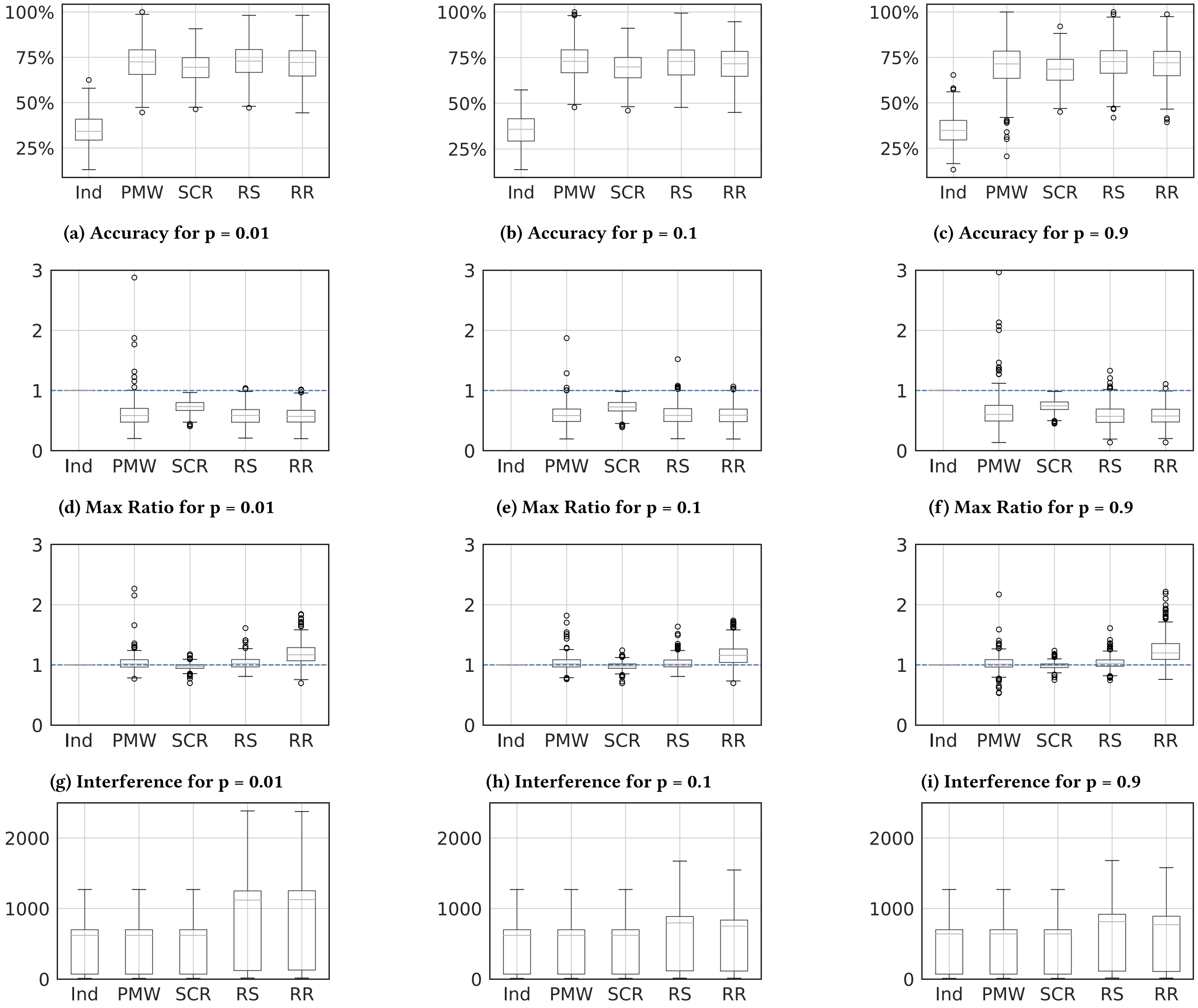
8.1 Experimental Setup

The following experiments are largely similar to the experiments in Section 4 which have been extended to the case of more than 2 analysts. We consider 10 analysts each with equal shares of the privacy budget $\epsilon/10$. This allows for more interactions and more complex interactions between analysts than in the 2 analyst case. We evaluated several different privacy budgets $\epsilon \in [0.1, 1, 10]$ but found that the results remained largely the same across privacy budgets. As such all the results shown below are using a moderate privacy budget $\epsilon = 1$.

A query sequence is generated by first assigning each analyst with a workload from a list of either one of the census race workloads [18], the identity workload, prefix sum, or H2 workload. Each of these workloads can either be asked on the entire database or a subset of the database defined by a predicate. This ensures that there is a mix of overlapping and disjoint queries. These workloads are then merged by randomly selecting an analyst to answer a query at each time step. Like in Section 4 there is a parameter p which denotes the probability that the first analyst is selected to answer a query. The first analyst is chosen with probability p each of the other analysts are chosen with probability $(1-p)/9$. In this case, since there are 10 analysts $p = 0.1$ corresponds to the case where an analyst is chosen uniformly at random each time and values of p above that signify that the first analysts asks their queries with higher probability. We include p values of 0.01 , 0.1, and 0.9. This includes a case when one analyst is underrepresented, a uniform distribution, and a case where one analyst is vastly over represented.

8.2 Mechanisms

We use **Independent PMW** (ind) as our baseline mechanism which we will compare our other mechanisms to. In this mechanism, each analyst is given their own independent instance of PMW and asks their queries exclusively on their respective instance. Since



(a) Accuracy for p = 0.01

(b) Accuracy for p = 0.1

(c) Accuracy for p = 0.9

(d) Max Ratio for p = 0.01

(e) Max Ratio for p = 0.1

(f) Max Ratio for p = 0.9

(g) Interference for p = 0.01

(h) Interference for p = 0.1

(i) Interference for p = 0.9

(j) Time Steps for p = 0.01

(k) Time Steps for p = 0.1

(l) Time Steps for p = 0.9

Figure 3: Accuracy (First row), Max Ratio Error (Second row), Empirical Interference (Third row), and Time to Completion (Last row) for the following five algorithms: Independent PMW(Ind), Joint PMW (PMW), Seeded Cache and Reconstruct (SCR), Round Robin PMW (RR), and Randomized PMW (RS). p represents the probability that the first analyst (among 10) has her queries answered at any time step.

this mechanism does not allow any interaction between analysts it satisfies the sharing incentive, analyst monotonicity, and non-interference.

We also compare our mechanisms against a mechanism that is designed to optimize for overall utility without any regard for the desiderata. For this purpose, we also evaluate **Private Multiplicative Weights** (PMW) as a general online differentially private mechanism, without regard to analyst identity.

The first of our proposed mechanisms is **Seeded Cache and Reconstruct** (SCR), explained in detail in Section 6. In addition, we test the efficacy of both the randomized scheduler and the round robin scheduler. For both schedulers, we create a single instance of PMW and pass that into the scheduler as a parameter. As a result, we evaluate **Round Robin PMW** (RR) and **Randomized Scheduler PMW** (RS) as instances of the round robin scheduler and randomized scheduler respectively.

8.3 Empirical Measures

We measure overall utility as the number of queries that can be answered with error under a threshold α. For each analyst, their individual utility is measured as the number of queries belonging to that analyst that are answered with error under threshold α.

In addition to utility, we measure the Maximum Ratio Error and Empirical Interference as in [20]. These are measures of how severe any violations of the sharing incentive and non-interference respectively. The ratio error of a mechanism $\mathcal{M}$ of a given analyst i is the utility of $\mathcal{M}$ in the independent case divided by the utility of the $\mathcal{M}$ in the joint case. This value measures the sharing incentive and to what extent it is violated. Values greater than 1 signify a violation of the sharing incentive and larger values signify larger violations. Here we present the **Maximum Ratio Error** which is the maximum of all ratio errors taken across all analysts.

$$\max_i \left(\frac{U_i(\mathcal{M}, Q^i, s_i\epsilon)}{U_i(\mathcal{M}, Q, \epsilon)} \right)$$

The Empirical Interference is a measure of the extent to which a mechanism violates non-interference. For any analyst i the interference with respect to another analyst j is measured as the ratio of utility of analyst i under mechanism $\mathcal{M}$ excluding j and the utility of analyst i under the same mechanism with all analysts present. Like before if this ratio is larger than 1 then analyst i experiences more utility when j is excluded than when they are included in the mechanism, indicating a violation of non-interference. We define the Empirical Interference as the maximum interference across any pair of analysts as follows.

$$\max_{i,j,i \neq j} \frac{U_i(\mathcal{M}, Q \setminus Q^j, (1 - s_j)\epsilon)}{U_i(\mathcal{M}, Q, \epsilon)}$$

The Query Schedulers incur an additional cost in that they can stall when queries are not available. In order to measure the impact of the schedulers' stalling, we measure the Time To Completion, the number of time steps necessary for a mechanism to answer all the queries. For the non-scheduled mechanisms this will simply measure the size of the query sequence but for the scheduled mechanisms this will capture the stalling time in addition to the time answering the queries.

8.4 Results

We see in Figure 3 the results for $\epsilon = 1$ and $\alpha = 0.01$.

8.4.1 Utility. In terms of utility, it is clear that independently answering queries results in a severe decrease in utility. Independent PMW can answer less than half of the total queries under the threshold while all of the other mechanisms perform significantly better under all values of p.

While SCR is analyst monotonic in all cases it comes at a slight cost. SRC consistently answers slightly fewer queries than the optimal PMW or any of the schedulers but significantly better than independent mechanisms. Since the schedulers only re-order the query sequence they perform nearly identically to PMW. They even outperform PMW in pathological cases where one analyst's queries are over-represented at the beginning of the sequence when $p = 0.9$.

8.4.2 Max Ratio Error. PMW regularly violates the sharing incentive for all values of p. This is particularly severe in cases where one analyst is either over or under represented. In those cases, some analysts can see as many as 3 times more queries answered in the independent case as opposed to the joint case. SCR has no violations of the sharing incentive in any case whereas the schedulers only observe violations in outlier cases due to the inherent randomness of those mechanisms.

8.4.3 Empirical Interference. In most cases, PMW sees a violation of non-interference, with only rare cases having no violations. Both schedulers also violate non-interference in most cases and the violation grows as p increases. Of the two schedulers, the randomized scheduler observes significantly less severe violations of the sharing incentive. For example in Figure 3i in the worst case, an analyst can only answer 1.5× the queries without one analyst whereas the round robin scheduler sees a 2.2× difference in the worst case. SCR sees only a few outlier violations of the sharing incentive which can be attributed to randomness. Unlike the schedulers, which do not provably satisfy the sharing incentive, SCR sees only minor violations and still satisfies the sharing incentive in expectation.

8.4.4 Time to Completion. Time to completion remains the same for all the non-schedulers as there is no possibility for stalling. In these cases, the time to completion is simply the number of queries in the sequence. In all cases, both schedulers incur some penalty in time to completion. This is the most extreme in the case of $p = 0.01$ when one analyst is severely underrepresented. This can lead to frequent stalling where the mechanism is waiting on the underrepresented analyst. In the worst case, this leads to the schedulers taking up to twice as many iterations to completely answer the entire sequence.

8.5 Discussion

We demonstrated that both seeded cache and reconstruct and the query schedulers are efficient solutions that both satisfy the sharing incentive. While they are both viable mechanisms each serves its own purpose and the choice of which to use is left to the data curator. Seeded cache and reconstruct has the benefit of being provably analyst monotonic and as such can be used in high stakes cases where satisfying the criteria is crucial. The query schedulers, however, have the benefit that they can be built on top of existing state of the art mechanisms and are not subject to the fundamental query limit. These can be used in lower stakes cases where the additional utility from the current state of the art outweighs the need for the guarantees of non-interference.

9 CONCLUSION

We demonstrate through Theorem 6 that online mechanisms can either answer a large number of queries (like PMW) or satisfy the sharing incentive when the ordering is possibly adversarial. This result relies heavily on the ability for queries to arrive in an adversarial order where one analyst monopolizes those first set of queries. We first propose Seeded Cache and Reconstruct, a mechanism which is analyst monotonic in all cases but is subject to the limit of Theorem 6. We then propose the alternative Query Scheduler which allows existing state-of-the-art online query answering mechanisms to satisfy the sharing incentive without compromising performance.

REFERENCES

[1] Mark Bun, Thomas Steinke, and Jonathan R. Ullman. 2016. Make Up Your Mind: The Price of Online Queries in Differential Privacy. *CoRR* abs/1604.04618 (2016). arXiv:1604.04618 http://arxiv.org/abs/1604.04618

[2] Bolin Ding and Marianne Winslett. 2011. Differentially Private Data Cubes : Optimizing Noise Sources and Consistency. (2011).

[3] Irit Dinur and Kobbi Nissim. 2003. Revealing information while preserving privacy. *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS 03* (2003). https://doi.org/10.1145/773153.773173

[4] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer Berlin Heidelberg.

[5] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the Third Conference on Theory of Cryptography* (New York, NY) *(TCC'06)*. Springer-Verlag, Berlin, Heidelberg, 265–284. https://doi.org/10.1007/11681878_14

[6] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* (2014).

[7] Gagan Goel and Aranyak Mehta. 2008. Online Budgeted Matching in Random Input Models with Applications to Adwords. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, California) *(SODA '08)*. Society for Industrial and Applied Mathematics, USA, 982–991.

[8] Gagan Goel and Aranyak Mehta. 2008. Online Budgeted Matching in Random Input Models with Applications to Adwords. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, California) *(SODA '08)*. Society for Industrial and Applied Mathematics, USA, 982–991.

[9] Anupam Gupta and Sahil Singla. 2020. Random-Order Models. *CoRR* abs/2002.12159 (2020). arXiv:2002.12159 https://arxiv.org/abs/2002.12159

[10] Samuel Haney, Ashwin Machanavajjhala, John M. Abowd, Matthew Graham, Mark Kutzbach, and Lars Vilhuber. 2017. Utility Cost of Formal Privacy for Releasing National Employer-Employee Statistics. In *Proceedings of the 2017 ACM International Conference on Management of Data* (Chicago, Illinois, USA) *(SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 1339–1354. https://doi.org/10.1145/3035918.3035940

[11] Moritz Hardt and Guy N. Rothblum. 2010. A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis. In *FOCS*. IEEE Computer Society, 61–70. https://doi.org/10.1109/FOCS.2010.85

[12] Moritz Hardt and Kunal Talwar. 2009. On the Geometry of Differential Privacy. arXiv:0907.3754 [cs.CC]

[13] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment* (2010).

[14] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. 2010. Optimizing Linear Counting Queries Under Differential Privacy. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Indianapolis, Indiana, USA) *(PODS '10)*. ACM, New York, NY, USA, 123–134. https://doi.org/10.1145/1807085.1807104

[15] Chao Li and Gerome Miklau. 2013. Optimal Error of Query Sets Under the Differentially-private Matrix Mechanism. In *Proceedings of the 16th International Conference on Database Theory (ICDT '13)*. ACM.

[16] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. 2008. Privacy: Theory meets Practice on the Map. In *2008 IEEE 24th International Conference on Data Engineering*. 277–286. https://doi.org/10.1109/ICDE.2008.4497436

[17] John Abowd Daniel Kifer Brett Moran Robert Ashmead Philip Leclerc Wiliam Sexton Simson Garfinkel Ashwin Machanavajjhala. [n.d.]. Census TopDown: Differentially Private Data, Incremental Schemas, and Consistency with Public Knowledge. ([n. d.]). https://columbia.github.io/private-systems-class/papers/Abowd2019Census.pdf

[18] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2018. Optimizing Error of High-dimensional Statistical Queries Under Differential Privacy. *PVLDB* 11, 10 (2018).

[19] Donald J. Newman. 1960. The Double Dixie Cup Problem. *The American Mathematical Monthly* 67, 1 (1960), 58–61. http://www.jstor.org/stable/2308930

[20] David Pujol, Yikai Wu, Brandon Fain, and Ashwin Machanavajjhala. 2021. Budget Sharing for Multi-Analyst Differential Privacy. *Proc. VLDB Endow.* 14, 10 (2021), 1805–1817. http://www.vldb.org/pvldb/vol14/p1805-pujol.pdf

[21] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding Hierarchical Methods for Differentially Private Histograms. *Proc. VLDB Endow.* 6, 14 (Sept. 2013), 1954–1965. https://doi.org/10.14778/2556549.2556576

[22] Nathan B. Shank and Hannah Yang. 2013. Coupon collector problem for non-uniform coupons and random quotas. *The Electronic Journal of Combinatorics* 20, 2 (2013). https://doi.org/10.37236/3348

[23] Jaideep Vaidya, Basit Shafiq, Xiaoqian Jiang, and Lucila Ohno-Machado. 2013. Identifying inference attacks against healthcare data repositories. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science* (Mar 2013). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3845790/

[24] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. 2014. Towards Accurate Histogram Publication under Differential Privacy. *Proc. SIAM SDM Workshop on Data Mining for Medicine and Healthcare* (2014).